3. Apply Functions

Note on "...": Each of our graphing functions uses a "..." parameter to receive arguments to be passed on to `par()` (e.g., `?plot`). We can use "..." too. e.g. Here's a silly wrapper function.

```
red.density = function(x, ...) { # (Try it without "..." parameter too.)
  plot(density(x), col="red", ...) # Pass extra arguments to plot().
}
red.density(x=rnorm(100), main="Two extra arguments", lty="dashed")
```

The `apply` family of functions makes many explicit loops unnecessary by applying a function, passed via the parameter `FUN`, to each value in a vector (or some other data structure). In each `apply` function's parameter list, "..." refers to extra arguments passed to `FUN`.

- `lapply(X, FUN, ...)` ("list apply") applies function `FUN` to each element of vector or list `X`, returning a list of the same length as `X`. e.g.

  ```
  (average = lapply(X=mtcars, FUN=mean)) # (treating data frame as a list of vectors)
  ```

- `sapply(X, FUN, ...)` ("simplified apply") is a wrapper for `lapply(X, FUN, ...)` that returns a vector, matrix, or array instead of a list. e.g.

  ```
  sapply(X=mtcars, FUN=mean)
  sapply(X=mtcars, FUN=quantile)
  ```

- `mapply(FUN, ...)` ("multiple arguments apply") takes the several vectors in `...` and applies `FUN` to all first elements, then to all second elements, etc. e.g.

  ```
  x = 1:4
  y = 5:8
  z = 9:12
  mapply(sum, x, y, z)
  ```

- `apply(X, MARGIN, FUN, ...)` runs `FUN` on "margins" of array `X`, keeping those dimensions specified in `MARGIN`. e.g.

  ```
  m = matrix(data=1:6, nrow=2, ncol=3)
  apply(X=m, MARGIN=1, FUN=sum) # keep dimension 1 (rows)
  apply(X=m, MARGIN=2, FUN=sum) # keep dimension 2 (columns)
  ```

- `tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)` applies `FUN` to a subset of vector `X` for each combination in the list of factors `INDEX` (each having the same length as `X`). e.g.

  ```
  tapply(X=mtcars$mpg, INDEX=mtcars$cyl, FUN=mean)
  tapply(X=mtcars$mpg, INDEX=list(mtcars$cyl, mtcars$gear), FUN=mean)
  mtcars[(mtcars$cyl==6) & (mtcars$gear==3), ] # check tapply() output

  # use ... to pass extra argument "probs=c(.25, .75))" to FUN=quantile:
  tapply(X=mtcars$mpg, INDEX=mtcars$cyl, FUN=quantile, probs=c(.25, .75))
  ```